Event Management Pro v5 - PHP Demo Application

PHP >=7.4 (8.0+ Recommended) DB MySQL | SQLite License MIT

Overview

Event Management Pro v5 is a demonstration web application built with plain PHP for managing events, user authentication, and RSVPs. It uses PDO for database interaction (supporting MySQL & SQLite) and incorporates basic security practices.

This README provides specific instructions on how to set up and run this project locally using popular development environments like **XAMPP** or **WAMP**.

Note: This is a demonstration project and requires further development and security hardening for production use.

Features

- User Registration & Login (Admin/User Roles)
- Password Hashing
- Admin Event Management (Create, Read, Update, Delete)
- User Event RSVP & Cancellation System (with capacity checks)
- Session Management
- CSRF Protection & Output Escaping
- PDO Database Interaction (MySQL/SQLite compatible)
- Basic Templating & Frontend Styling
- Automated Setup Script (`scripts/setup.sh`) - *Optional alternative setup described below*

Technology Stack

- **Backend:** PHP (7.4+)
- **Database:** MySQL (MariaDB) or SQLite 3
- **Frontend:** HTML5, CSS3, JavaScript (ES6+)
- **Local Server:** XAMPP / WAMP (or similar like MAMP, Laragon)

Setup and Installation (XAMPP / WAMP)

Follow these steps to get the project running on your local machine using XAMPP or WAMP.

Prerequisites:

- XAMPP or WAMP installed and running (ensure Apache and MySQL services are started).
- A web browser.
- Access to phpMyAdmin (usually included with XAMPP/WAMP).
- A text editor or IDE.

Steps:

1. **Download or Clone Project:**
    - Obtain the project files (e.g., download a ZIP or clone if it's in a Git repository).
    - Extract/place the entire project folder (e.g., `event_management_pro`) into the web server's document root directory:
        - **XAMPP:** `C:\xampp\htdocs\` (or `/Applications/XAMPP/htdocs/` on macOS)
        - **WAMP:** `C:\wamp\www\` or `C:\wamp64\www\`
        - You should now have a path like `C:\xampp\htdocs\event_management_pro\`
2. **Configure `config.php` :**
    - Navigate to the `includes` directory within your project folder (e.g., `htdocs\event_management_pro\includes`).
    - Open `config.php` in your text editor.
    - **Choose Database Type:**
        - Set `define('DB_TYPE', 'mysql');` (Recommended for XAMPP/WAMP default setup).
        - *Alternatively, you could use 'sqlite', but the setup below focuses on MySQL.*
    - **Configure MySQL Settings:**
        - `define('DB_HOST', '127.0.0.1');` (or `localhost`) - Usually correct for XAMPP/WAMP.
        - `define('DB_NAME', 'event_mgmt_pro_db');` - Choose a name for your database. You will create this in the next step.
        - `define('DB_USER', 'root');` - Default XAMPP/WAMP MySQL username.
        - `define('DB_PASS', '');` - Default XAMPP/WAMP MySQL password is often empty. **Set a password in a real environment!**
    - **Verify `BASE_URL` :**
        - Check if the auto-detected `BASE_URL` looks correct for your local setup (e.g., `http://localhost/event_management_pro/public/`).
        - If not, **uncomment** the manual definition line and set it explicitly:

            ```
            // Example: define('BASE_URL', 'http://localhost/event_management_pro/public/');
            ```

            Make sure it points to the `public` directory and ends with a slash (`/`).
    - Save the changes to `config.php` .
3. **Create and Import the Database (MySQL using phpMyAdmin):**
    - Open your web browser and go to `http://localhost/phpmyadmin` .

- **Create Database:**
  - Click on the "Databases" tab (or "New" on the left sidebar).
  - In the "Create database" field, enter the exact database name you set in `config.php` (e.g., `event_mgmt_pro_db`).
  - Choose a collation like `utf8mb4_unicode_ci`.
  - Click "Create".
- **Import Schema:**
  - Click on the newly created database name in the left sidebar to select it.
  - Click on the "Import" tab at the top.
  - Click the "Choose File" or "Browse..." button.
  - Navigate to your project folder and go into the `scripts/` directory.
  - Select the `mysql_schema.sql` file. *(**Note:** This file should have been generated by the original script or provided with the project. If not, you might need to run `scripts/setup.sh` in a bash environment first, or manually execute the SQL commands found within `scripts/initialize_db.php`).*
  - Ensure the character set is `utf8mb4`.
  - Click the "Go" or "Import" button at the bottom. You should see success messages indicating tables (`users`, `events`, `rsvps`) were created.
- **(Optional) Import Sample Data:**
  - If you want sample users and events, you can manually execute the `INSERT` statements found within the `scripts/insert_sample_data.php` file. Go to the "SQL" tab in phpMyAdmin (while your database is selected) and paste/run the relevant SQL insert commands.

4. **(Alternative) Using the Setup Script (Requires Bash Environment):**
   - If you have a bash environment available (like Git Bash on Windows, or WSL, or native Linux/macOS), you can use the provided setup script as an alternative to manual database creation/import.
   - Ensure PHP CLI is available in your bash environment.
   - Navigate to the `scripts` directory in your terminal: `cd /path/to/htdocs/event_management_pro/scripts`
   - Make the script executable (if needed): `chmod +x setup.sh`
   - Run the script: `./setup.sh`
   - This script will read `config.php`, connect to MySQL, attempt to create the database and tables, and optionally insert sample data based on your prompts. **Ensure your MySQL service is running before executing.**
5. **Run the Project:**
   - Ensure Apache and MySQL services are running in your XAMPP/WAMP control panel.
   - Open your web browser.
   - Navigate to the `public` directory of your project via localhost. The URL should match the `BASE_URL` you verified/set: `http://localhost/event_management_pro/public/`
   - You should see the Event Management Pro homepage.

Usage

- Navigate the site to view events.
- Register for a user account (`/public/register.php`).
- Login (`/public/login.php`). Sample logins (if sample data imported):
  - Admin: `admin` / `adminpass`
  - User: `testuser` / `userpass`
  - **(Change default passwords!)**
- RSVP for events (as a user).
- View your RSVPs on the dashboard (`/public/dashboard.php`).
- Manage events via the admin panel (`/public/admin.php` - requires admin login).

troubleshooting

- **`Access denied for user 'root'@'localhost'`**: Check the `DB_PASS` in `config.php`. XAMPP/WAMP default is often empty, but if you've set a MySQL root password, enter it there.
- **`Unknown database 'event_mgmt_pro_db'`**: Ensure you created the database in phpMyAdmin (Step 3) with the exact name specified in `config.php`.
- **`Table '...' doesn't exist`**: Make sure the database import (Step 3) was successful and created the `users`, `events`, and `rsvps` tables. Check phpMyAdmin. If using the setup script, review its output for errors.
- **PHP Errors Displayed / Blank Page:**
  - Check the Apache error log (`C:\xampp\apache\logs\error.log` or similar in WAMP).
  - Check the PHP error log (`C:\xampp\php\logs\php_error_log` or enabled via `php.ini`).
  - Check the project's `data/php_errors.log` (if configured).
  - Ensure the required PHP extensions (PDO, pdo_mysql/pdo_sqlite) are enabled in your `php.ini` file (accessible via XAMPP/WAMP control panel).
- **`Not Found` Errors:** Double-check the `BASE_URL` in `config.php` and ensure the URL you are visiting corresponds correctly to the `public` directory (e.g., `http://localhost/project_folder/public/`). Ensure `mod_rewrite` is enabled in Apache if you encounter issues related to URL routing (though this project doesn't heavily rely on it initially).
- **Permission Errors (SQLite):** If using SQLite, ensure the `data/` directory within your project in `htdocs` / `www` is writable by the Apache process. Setting permissions in Windows can be tricky; sometimes running XAMPP/WAMP as Administrator can help (for local development only).

Event Management Application - Development Evolution (v1-v3)

Overview

This document outlines the development journey of the Event Management Application, showcasing its incremental evolution across three major versions (v1, v2, v3).

The project progressed from a basic concept to a more structured, feature-rich, and secure web application designed for managing events and handling user interactions.

- **Version 1:** Focused on establishing the foundational concept and basic functionality.
- **Version 2:** Shifted towards improved code structure and acting primarily as an event "showcasing" platform.
- **Version 3:** Matured into a professional, interactive platform incorporating industry-standard security practices, user authentication, event booking/RSVP capabilities, flexible database support, and an automated setup process.

Version Evolution

Version 1: The Foundation

- **Goal:** Proof-of-concept, basic event listing.
- **Functionality:**
  - Display a static or minimally dynamic list of events.
  - Very basic page structure.
- **Code Structure:**
  - Likely monolithic PHP files or minimal includes.
  - Primarily procedural code.
  - Limited separation of concerns (HTML, CSS, PHP potentially mixed).
  - Basic database connection (if any), possibly using older methods (e.g., `mysql_*` - deprecated).
- **Security:** Minimal attention to security practices (e.g., no CSRF protection, potential SQL injection vulnerabilities, no output escaping).
- **Setup:** Fully manual (copying files, creating database tables manually).

Version 2: The Showcase & Structural Improvement

- **Goal:** Enhance presentation, improve code organization, act as a display/portfolio page for events.
- **Functionality:**
  - Improved event listing and display (better styling, potentially basic filtering).
  - Focus on showcasing event information rather than interaction (booking/RSVP might be absent or very basic).
  - May include a simple way to update displayed event data (e.g., editing a data file or a very basic admin interface).
- **Code Structure:**
  - **Incremental Updates:** Introduction of more `include` files for reusable components (header, footer).
  - Basic helper functions might be introduced.
  - Slightly better separation of PHP logic and HTML presentation.
  - Database interaction might be more organized, perhaps using early PDO concepts but not fully structured.
- **Security:**
  - **Incremental Updates:** Basic output escaping (`htmlspecialchars`) might be introduced to prevent simple XSS.
  - Still lacking comprehensive security measures found in v3.
- **Setup:** Still largely manual, but file organization might be slightly improved.

Version 3: The Professional Platform (Event Management Pro v5)

- **Goal:** Create a functional, interactive, and secure platform for managing events, user registrations, and RSVPs, adhering to better development practices and offering easier setup.
- **Functionality:**
  - **Full User Authentication:** Secure Registration, Login (Password Hashing), Logout, Role-based access (Admin vs User).
  - **Event CRUD:** Admins can Create, Read, Update, and Delete events via a dedicated panel.
  - **RSVP System:** Users can RSVP for events, cancel RSVPs, with checks against event capacity.
  - **User Dashboard:** Users see events they have RSVP'd for.
  - **Admin Panel:** Centralized event management.
  - **Dynamic Event Listing & Details:** Displays upcoming/ongoing events, shows detailed views.
  - **Flash Messages:** User feedback for actions (success, error, info).
- **Code Structure:**
  - **Modular Design:** Clear separation into `public/` , `includes/` , `data/` , `scripts/` directories.
  - **Core Includes:** Dedicated files for `config.php` , `database.php` (PDO connection, session start), `functions.php` (reusable helpers).
  - **Templating:** Simple but effective PHP templating (`renderTemplate` , partials like `header` , `footer` , `navbar`).
  - **Database Abstraction:** Consistent use of **PDO** with **prepared statements**.
- **Industry Standard Security Practices:**
  - **CSRF Protection:** Implemented on all state-changing forms.
  - **Password Hashing:** Uses `password_hash()` and `password_verify()` .
  - **SQL Injection Prevention:** Relies entirely on PDO prepared statements.
  - **XSS Prevention:** Consistent output escaping (`htmlspecialchars`).
  - **Secure Sessions:** Configurable secure cookie parameters (HttpOnly, Secure, SameSite).
  - **Directory Protection:** `.htaccess` rules prevent direct access to non-public directories.
- **Database Flexibility & Schema:**
  - **Dual DB Support:** Designed to work seamlessly with both **MySQL** and **SQLite** via configuration toggle (`config.php`).
  - **Well-Defined Schema:** Clear structure for `users` , `events` , `rsvps` tables with appropriate relationships and constraints (Foreign Keys, `ON DELETE CASCADE`).
  - **Schema Generation:** `initialize_db.php` creates the schema, and `mysql_schema.sql` is generated for reference/manual import if using MySQL.
- **Automated Setup & Ease of Use:**
  - **`setup.sh` Script:** Automates the database initialization process (table creation) based on `config.php` .
  - **Optional Sample Data:** Script can populate the database with initial users and events for quick testing.
  - **Clear Instructions:** Setup script provides guidance on configuration and next steps.
- **Technology:** Mature PHP practices (strict types, PDO), structured CSS, minimal vanilla JavaScript for UI enhancements.

Technology Stack (v3)

- **Backend:** PHP (7.4+, 8.0+ Recommended)
- **Database:** PDO supporting MySQL/MariaDB and SQLite 3
- **Frontend:** HTML5, CSS3, JavaScript (ES6+)
- **Setup:** Bash Scripting

Setup and Installation (v3)

Version 3 provides an automated setup process:

1. **Configure:** Edit `includes/config.php` to set `DB_TYPE` ('mysql' or 'sqlite') and corresponding database credentials or path. **Crucially, change default passwords if using MySQL sample data.**
2. **Run Setup Script:**

   ```
   cd event_management_pro/scripts
   ./setup.sh
   ```

   This script handles prerequisite checks, directory setup, database table creation, and optionally inserts sample data.
3. **Web Server Config:** Point your web server's document root to the `event_management_pro/public` directory. Ensure PHP is configured correctly.
4. **Permissions:** Ensure the `data/` directory is writable by the web server, especially for SQLite.

(Refer to the detailed v3 README for more specific setup instructions)

Project Structure (v3)

(The v3 structure as detailed in the previous README)

Usage (v3)

- **Visitors:** View events.
- **Users:** Register, Login, View event details, RSVP/Cancel, View Dashboard.
- **Admin:** Login, Full Event CRUD, View all events.

Security Focus (v3)

Version 3 significantly elevates security by implementing:

- CSRF Protection
- Password Hashing (bcrypt)
- Prepared Statements (PDO)
- Output Escaping
- Secure Session Cookies
- Directory Access Control

Database Flexibility (v3)

Version 3 leverages PDO to provide native support for both **MySQL** and **SQLite** databases, selectable via a simple configuration setting. The setup script handles the schema creation for the chosen database type.

Conclusion

The Event Management Application demonstrates a clear progression in web development practices. Starting from a basic concept (v1), it evolved through structural improvements and a focus on presentation (v2), culminating in a functional, interactive, and significantly more secure platform (v3) that incorporates industry-standard practices, automated setup, and database flexibility suitable for a professional demonstration or a strong starting point for further development.

Event Management Pro v5 - SithumSithara RUSL Applied Science Web Project

Overview

This repository contains **Event Management Pro v5**, a web application developed as part of the **SithumSithara RUSL Applied Science Web Project**. Built with plain PHP, it demonstrates core web development principles for managing events, user authentication, and RSVPs.

The project emphasizes foundational PHP skills, database interaction via PDO (supporting MySQL/SQLite), basic security practices, and a structured approach without relying on a full framework. It provides a practical example of event listing, user registration/login, admin event management (CRUD), and user RSVP functionality.

Context Note: While the current version focuses on core CRUD and RSVP features, a potential future direction or related project context could involve integrating **interactive charts** (e.g., using Chart.js or ApexCharts) to visualize data such as event attendance trends, registration numbers over time, or service popularity, adding an analytical dimension to the event management data.

Disclaimer: This is a demonstration or educational project and is **not suitable for production deployment** without significant security hardening, feature expansion, and code review.

Core Functionalities Implemented

This project showcases the following individual functionalities:

- **Backend Structure:**
  - **Modular Code:** Separation of concerns using `includes/` for configuration, database logic, helper functions, and templates.
  - **Configuration Management:** Centralized settings in `config.php` (database type, credentials, paths, session parameters).
  - **Database Abstraction (PDO):** Uses PHP Data Objects for database interaction, allowing easy switching between MySQL and SQLite.
  - **Helper Functions:** A library (`functions.php`) providing reusable functions for:
    - URL Generation (`baseUrl`)
    - Redirection (`redirect`)
    - Authentication Checks (`isAuthenticated`, `isAdmin`, `requireAuth`)
    - Session Management (`logoutUser`, secure cookie parameters)
    - Output Escaping (`escape` / `e` for XSS prevention)
    - Flash Messaging (`setFlashMessage`, `getFlashMessage`)
    - CSRF Protection (`generateCsrfToken`, `validateCsrfToken`)
    - Basic Template Rendering (`renderTemplate`)
    - Date Formatting (`formatDate`)
- **Database & Schema:**
  - **Relational Schema:** Defines tables for `users`, `events`, and `rsvps`.
  - **Foreign Key Constraints:** Enforces relationships (e.g., RSVPs linked to users and events) with cascading deletes.
  - **Automated Setup:** `scripts/setup.sh` orchestrates database initialization using `initialize_db.php`.
  - **Sample Data:** Optional script (`insert_sample_data.php`) to populate the database with initial users and events.
- **Authentication & Authorization:**
  - **User Registration:** Allows new users to create accounts (`register.php`).
  - **Password Security:** Uses `password_hash()` for secure password storage and `password_verify()` for comparison.
  - **Session-Based Login:** Manages user sessions securely after successful login (`login.php`).
  - **Logout:** Provides secure session termination (`logout.php`).
  - **Role Management:** Simple distinction between `admin` and `user` roles stored in the database and session.
  - **Access Control:** Protects routes/pages based on login status and user role (`requireAuth` function).
- **Event Management (Admin CRUD):**
  - **Create:** Admins can add new events with details like name, date, time, location, description, category, capacity, image URL (`event_edit.php`).
  - **Read:** Admins can view a list of all events (`admin.php`) and details of specific events (`event_view.php`).
  - **Update:** Admins can modify existing event details (`event_edit.php?id=...`).
  - **Delete:** Admins can remove events, automatically cascading to remove associated RSVPs (`event_delete.php`).
- **RSVP System (User Interaction):**
  - **RSVP:** Logged-in regular users can RSVP to 'Upcoming' or 'Ongoing' events (`rsvp.php` action='rsvp').
  - **Cancel RSVP:** Users can cancel their existing RSVP (`rsvp.php` action='cancel').
  - **Capacity Check:** Prevents users from RSVPing to events that have reached full capacity.
  - **Attendee Count:** Automatically increments/decrements the `attendees_count` on the `events` table within a transaction for data integrity.
  - **User Dashboard:** Users can view a list of events they have RSVP'd for (`dashboard.php`).
- **Frontend & UI:**
  - **Templating:** Simple PHP includes for header, footer, navigation, and message display.
  - **Basic Styling:** Uses CSS (`public/css/style.css`) for a dark theme, layout (Flexbox/Grid), basic component styling, and responsiveness.
  - **JavaScript Enhancements:** Minimal vanilla JS (`public/js/script.js`) for confirmation dialogs, auto-hiding flash messages, and active nav link highlighting.
  - **External Image Linking:** Supports linking event images via URL (field `image_url`).
- **Basic Security:**
  - **Directory Access Control:** `.htaccess` rules block direct web access to non-public directories.
  - **CSRF Protection:** Tokens generated and validated for all state-changing forms.
  - **XSS Prevention:** Output consistently escaped using `htmlspecialchars`.
  - **SQL Injection Prevention:** Uses PDO prepared statements exclusively for database queries.
  - **Secure Session Cookies:** Configured with `HttpOnly`, `Secure` (if HTTPS), and `SameSite` flags.

Technology Stack

- **Backend:** PHP (7.4+, 8.0+ Recommended)
- **Database:** PDO (PHP Data Objects) supporting:
  - MySQL / MariaDB
  - SQLite 3
- **Frontend:** HTML5, CSS3, JavaScript (ES6+)
- **Web Server:** Apache or Nginx (with mod_php/PHP-FPM) recommended.

Setup and Installation

(These steps remain the same as the previous README)

Prerequisites:

- PHP CLI (Command Line Interface) 7.4+ with PDO, pdo_mysql, and pdo_sqlite extensions enabled.
- A web server (Apache, Nginx, etc.).

- Access to a MySQL database server (if using MySQL).
- Bash environment (for running setup scripts).

Steps:

1. **Generate Project:** Use the `create_event_project.sh` script.
2. **Review Configuration (`event_management_pro/includes/config.php`):** Set `DB_TYPE`, credentials (especially DB_PASS), and `BASE_URL`.
3. **Run Setup Script:**

   ```
   cd event_management_pro/scripts
   ./setup.sh
   ```

   Follow prompts for database initialization and optional sample data.
4. **Configure Web Server:** Point DocumentRoot to `event_management_pro/public`. Ensure PHP processing is enabled.
5. **Set Permissions:** Ensure the `data/` directory (and potentially `data/php_errors.log`) is writable by the web server user, especially if using SQLite.
6. **Access Application:** Navigate to the configured URL in your browser.

Project Structure

(Structure remains the same as the previous README)

event_management_pro/ ├── data/ ├── includes/ │ ├── templates/ │ ├── config.php │ ├── database.php │ ├── functions.php │ └── .htaccess ├── public/ │ ├── assets/ │ │ ├── css/ │ │ ├── js/ │ ├── index.php │ ├── login.php │ └── ... (other public PHP files) ├── scripts/ │ ├── initialize_db.php │ ├── insert_sample_data.php │ ├── mysql_schema.sql │ ├── setup.sh │ └── .htaccess └── .gitignore

Usage

(Usage remains the same as the previous README)

- **Visitors:** View events on the homepage.
- **Users:** Register, Login, View event details, RSVP/Cancel RSVP, View their RSVPs on the dashboard.
- **Admin:** Login, Manage all events (Add, Edit, View List, View Detail, Delete).
  - *(Sample Data Logins if inserted)*: `admin` / `adminpass`, `testuser` / `userpass`.

Security Considerations

(Security points remain the same as the previous README)

- **DEMO ONLY:** Basic security implemented; not production-ready.
- Requires robust input validation, rate limiting, fine-grained permissions, etc., for real-world use.
- Review session and error handling configurations.

Future Enhancements

- **Interactive Charts:** Visualize data like RSVP counts per event, registration trends over time, or event popularity using a charting library (e.g., Chart.js, ApexCharts).
- Advanced Input Validation.
- Image Uploads (instead of URLs).
- Event Categories/Tags Filtering & Search.
- Email Notifications (Registration, RSVP, Reminders).
- Password Reset Functionality.
- AJAX for smoother UI interactions.
- Unit & Integration Testing.
- Production Deployment Configuration.

Project Context / Contributing

This project was developed for the SithumSithara RUSL Applied Science curriculum. While primarily an educational demo, feedback or suggestions related to the core concepts demonstrated are welcome.

License